Dynamic Form

This document gives information regarding the registration process followed in Regulatory Portal. These are the following steps in which a user can register themselves.

1. Create a user account by “create user” link in the application.
2. Confirm email id by clicking on the link.
3. Once confirmed, user can register themselves by filling out registration form.
4. Registration form has two parts for all the users.
5. Registration form’s first part is dynamic.

Currently dynamic form supports these controls

1. Input
2. Password
3. Button
4. DropDown
5. Email
6. FileUpload
7. AutoComplete
8. Gender
9. MultiSelect
10. DatePicker
11. Radio

We can also add other controls if required.

1. **How to Add controls in Registration Form (Controls supported by dynamic form).**
2. Insert record in UIControls table.

**Sample script**

**INSERT INTO UIControls (ControlOrder,PageID, ControlType, ControlName, Label, PlaceHolder, Subform, Mode, alignment, IsRequired)**

**VALUES(NULL, 8, 35,'policyIndemnityDocumentEndDate', 'Policy Indemnity End Date' , 'Policy Indemnity End Date' ,'InsurerRegistration', 'Add', 'left', 1)**

1. Arrange the order of UI Control by setting ControlOrder column. Make sure ControlOrder is consistent.

**2. How to Add controls in Registration Form (Controls supported by dynamic form).**

1. insert entry into dynamic dynamic-form.directive.ts

const components: { [type: string]: Type<Field> } = {

Radio: FormRadioButtonsComponent

Input: FormInputComponent,

Password: FormInputPasswordComponent,

Button: FormButtonComponent,

DropDown: FormSelectComponent,

Email: FormEmailComponent,

FileUpload: FormFileUploadComponent,

AutoComplete: FormAutocompleteComponent,

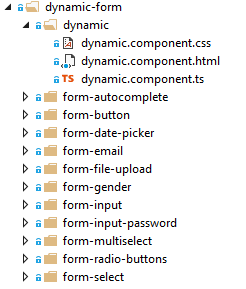
Gender: FormGenderComponent,

MultiSelect: FormMultiselectComponent,

DatePicker: FormDatePickerComponent,

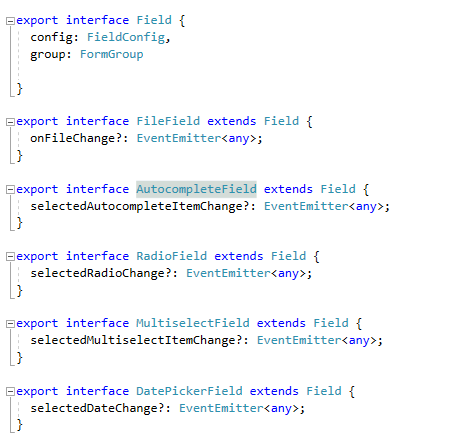
};

2. create a new folder in dynamic-folder. Follow the existing convention for creating folders for new dynamic components. A new control name must always start with “form-” as shown below.

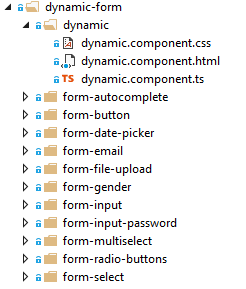


See form-autocomplete for reference.

1. Create a interface, in dynamic-form.directive.ts as below. This interface must extend existing interface “Field”



1. This interface will contain the output events which the new dynamic form going to emit or support. Please follow the existing naming convention while naming new interface and events for consistency.
2. This new interface must be extended by the new dynamic control you will going to add inside dynamic-form module.



Example is below



1. Add logic to create your dynamic component to view and listening to their events in dynamic-form.directive .ts
2. Creating component to view

}

else if (this.config.type == "AutoComplete") {

component = this.resolver.resolveComponentFactory<AutocompleteField>(components[this.config.type]);

this.component = this.container.createComponent<AutocompleteField>(component);

1. Listening to their events (optional, not required if you don’t want to listen any events of your new component.)

else if (this.config.type == "AutoComplete") {

this.component.instance.selectedAutocompleteItemChange.subscribe(message => {

this.onAutoCompleteItemChange.emit(message);

1. Write the logic to emit the events created in dynamic-directive.ts to the main parent page.
2. In dynamic.component.html , add the event properties. forms[0].config is for left aligned controls and forms[1].config is for right aligned control.

<div fxFlex="100" fxFlex.gt-xs="50" class="pr-1">

<div \*ngFor="let field of forms[0].config;"

appDynamicForm (onFileChanged)="handleFileChanged($event)"

***(onAutoCompleteItemChange)="handleAutocompleteChange($event)"***

(onSelectedRadioChange)="handleRadioChange($event)"

[config]="field"

[group]="group">

</div>

</div>

<div fxFlex="100" fxFlex.gt-xs="50" class="pr-1">

<div \*ngFor="let field of forms[1].config;"

appDynamicForm (onFileChanged)="handleFileChanged($event)"

***(onAutoCompleteItemChange)="handleAutocompleteChange($event)"***

(onSelectedRadioChange)="handleRadioChange($event)"

(onSelectedDateChange) ="handleSelectedDateChange($event)"

[config]="field"

[group]="group">

</div>

</div>

1. Add the following logic which emits the events to parent page in dynamic.component.ts file.

private handleAutocompleteChange(config) {

this.onAutocompleteItemChange.emit(config);

}

1. These new events needed to be listened in main parent page. Therefore, add the logic in parent html

<app-dynamic \*ngIf="forms"

[submitButtonText]="submitButtonText"

[includeDisableControlValue]="'true'"

[forms]="forms"

[listOfFiles]="lstFiles"

[mode]="mode"

#form="dynamicForm"

(onFileUpload)="onFileUploaded($event)"

(submit)="onClickSave($event)"

(onClickReset)="onClickReset()"

(onAutocompleteItemChange)="onAutocompleteItemChange($event)"

(onClickDeletedFile)="onClickDeletedFile($event)"

(onClickApprovedFile)="onClickApprovedFile($event)"

(onClickDeclincedFile)="onClickDeclincedFile($event)"

(onSelectedDateChanged)="onSelectedDateChanged($event)"

(onRadioItemChange) ="onRadioItemChange($event)"

>

</app-dynamic>

1. Now, add logic to listen to these events in parent page as below.

private onAutocompleteItemChange(config) {

let control = this.dynamicForm.group.get('visaPage');

if (config && config.value && control) {

if (this.localizationSettings.homeCountry.trim() != config.value[0].lookupDetailName.trim()) {

control.setValidators([Validators.required])

control.enable();

}

else {

control.clearValidators();

control.disable();

}

control.updateValueAndValidity();

}

}

The responsibility of populating dynamic controls with data rests with parent page. Therefore, all the data population logic must be added in registration.component.ts and not in dynamic controls.

For eg: in autocomplete, selection options must be loaded in registration.component.ts as below.

private getNationalityList() {

this.apiCallService.GetAll<LookupDetails[]>(ControllerUrlMapper.AutocompleteController + ActionUrlMapper.GetNationalityList).subscribe(response => {

if (this.configParts.find(control => control.name == "nationality" || control.name == "country")) {

this.nationalityControlType = this.configParts.filter(control => control.name == "nationality" || control.name == "country")[0];

this.nationalityControlType.options = response;

}

}

, error => {

this.alert([ErrorMessages.ErrorInternal]);

})

}

Additionally, in autocomplete, for selecting previously selected item(mostly, in Edit mode) must be added in registration.component.ts